

DeepCraft

Computer Vision in Minecraft

Justin

Mason

Miguel

Pato

Background



Objective

Compare convolutional neural networks (CNNs) to alternate classification methods in artificial 3D environments like Minecraft.


Will the same methods used to interpret real-life images work for Minecraft elements? Which method will perform best?

Goals:

- Biome Recognition
- Object Identification and Recognition
- Scene Recognition
- Deep Dreams



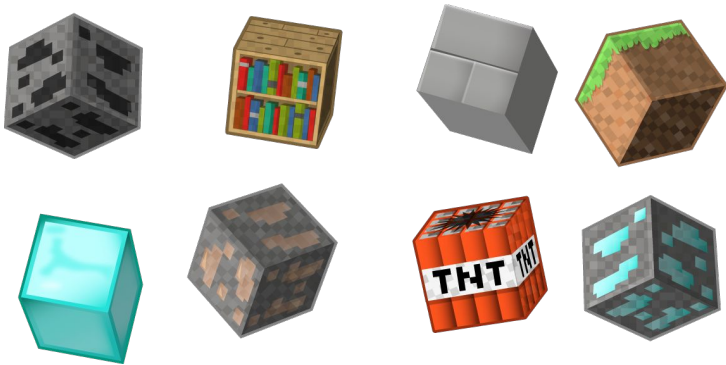
Literature

- ImageNet Classification with Deep CNNs
 - Creation of a CNN for image recognition
 - Rich feature hierarchies for accurate object detection and semantic segmentation
 - R-CNN for semantic segmentation
 - Fully Convolutional Networks for Semantic Segmentation
 - Image parsing: not ideal because of the training data needed
 - Inceptionism: Going Deeper into Neural Networks
 - Google's deep dreams
 - Other documentation: Caffe + Tensorflow
- 

Data Collection

The World of Minecraft

- 16x16 pixel blocks
- Focus on color
- Hard edges



Challenge: Capture and Label Objects



Solution: Use Video



Solution: Frames from Video



Extract one frame every two seconds ($\frac{1}{2}$ FPS)



Processing Steps

1. Convert the video to frames with ffmpeg
 - a. Select one frame every two seconds
 - b. Enforce that all videos are 256x256
2. Organize frames into folders representing their video name and labels
3. Manually discard any poor quality data
4. Select training and testing data with a custom Python script
 - a. Shuffle the videos
 - b. For each label, select distinct sets of training and testing videos
 - i. This prevents training and overfitting on highly correlated data from a single video
 - c. Save selections in a language-independent format (text file)



Data Set

Objects (9 Classes)

- Bunny, chicken, cow,
- cactus, pig, sheep, tree
- Forest (no object)
- Desert (no object)

- ~250 images per class
- 2000+ images

Biomes (4 Classes)

- Forest
- Desert
- Tundra
- Manmade

- 1500+ images per class



Classification

Object Classification: Unfiltered Images

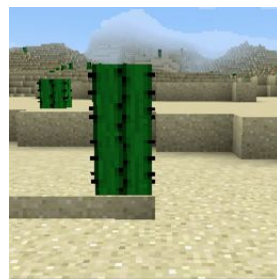
SVM as classifier

RGB and Grayscale

Single Class Accuracy: ~55-60%

Multi (7) Class Accuracy: ~18%

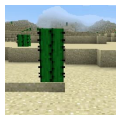
Single Class



Vs.



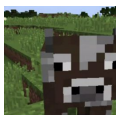
Multi Class



Vs.



Vs.



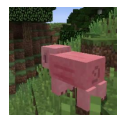
Vs.



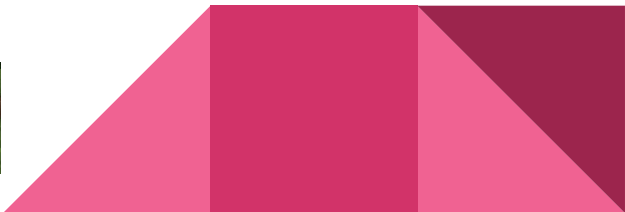
Vs.



Vs.



Vs.



Object Classification: Gabor Filters

Double Opponent Color Filters

Single Class Accuracy: 100%

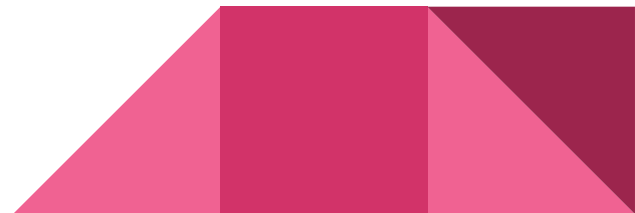
Multi Class Accuracy: 96%



red-green filtered cactus

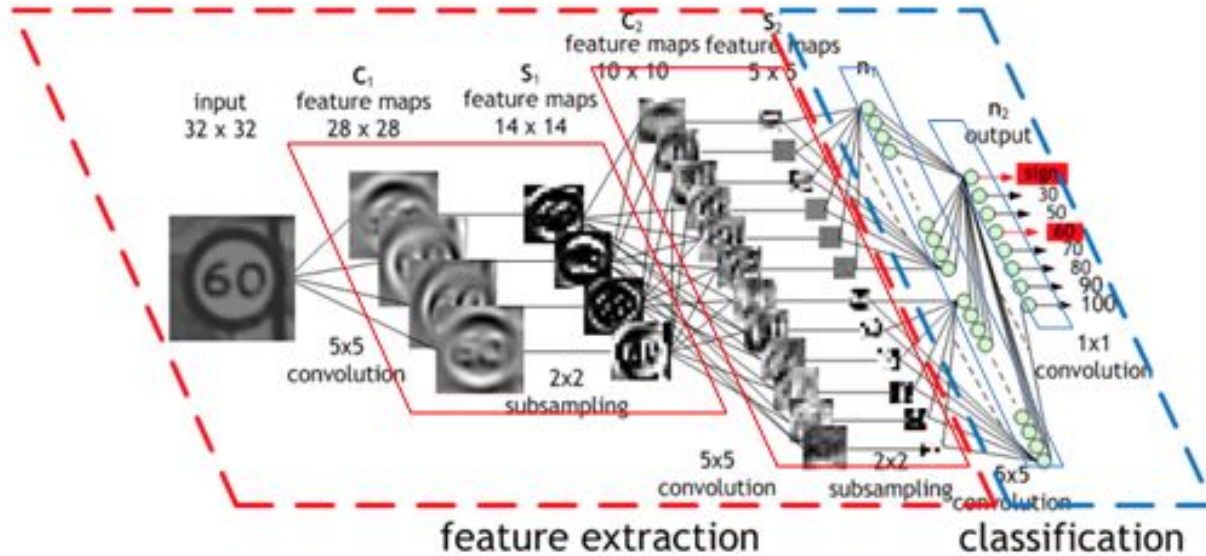


blue-yellow filtered sheep



Convolutional Neural Networks

MatConvNet, Caffe, and TensorFlow



MatConvNet (Pre-Trained CNN) - Biomes

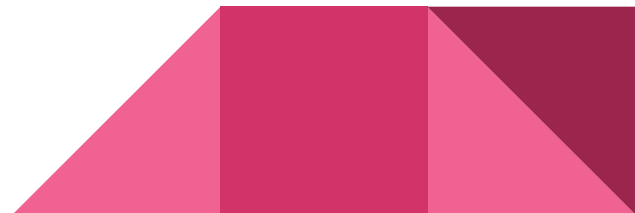
- MATLAB toolbox for CNNs
- Using the same pre-trained CNN and process from Homework #7

Results

- 100% Accuracy on Biomes

Tweaks

- Used grayscale to avoid classifying on color
- 98% accuracy



MatConvNet (Pre-Trained CNN) - Objects

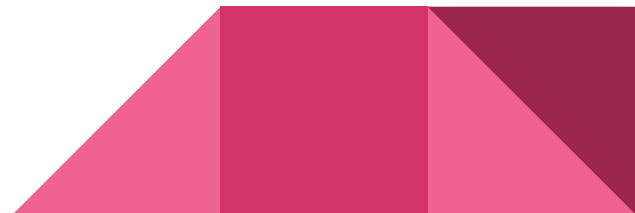
- Using the same pre-trained CNN and process from Homework #7

Results

- 81% Accuracy on 10 classes

Tweaks

- Used grayscale to check color dependence
 - 67% accuracy
- Excluded anti-objects in classification
 - 75% in rgb, 58% in gray.
 - Anti-objects proved to be extremely useful



TensorFlow (Our own CNN)

- Google's data flow graph framework in Python and C++
- Used to build a custom CNN from scratch

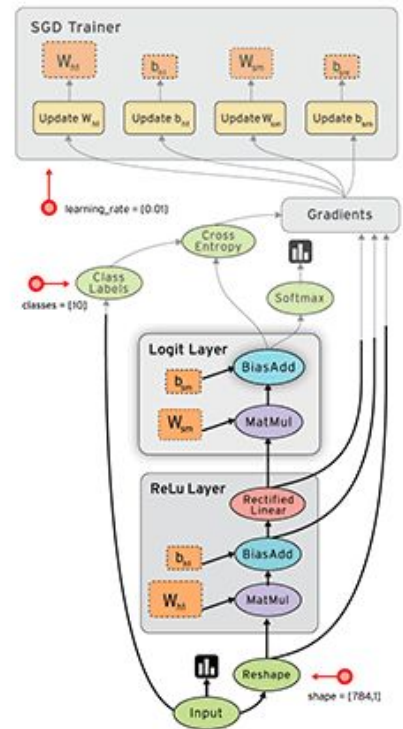
Structure:

3 layers of convolution -> rectification -> 2x2 maxpool -> dropout

1 fully connected layer -> dropout -> softmax classification

Roughly follows the structure of TensorFlow's AlexNet adaptation

Requires over 20M multiply and add operations per image



TensorFlow Results

Using 28x28 pixel images (downsampled with antialiasing)

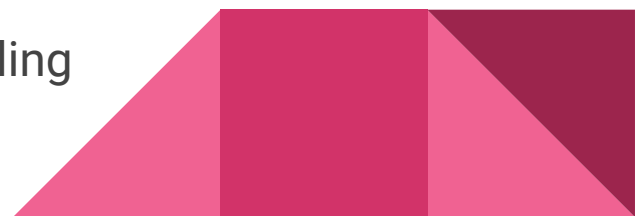
- Cactus vs desert (no object) - 98.6% accuracy (500 training steps)
- All 9 classes - 78.4% accuracy (500 training steps)
- All 9 classes - 80.9% accuracy (1000 training steps)

Using 128x128 pixel images lowered accuracy and increased runtime

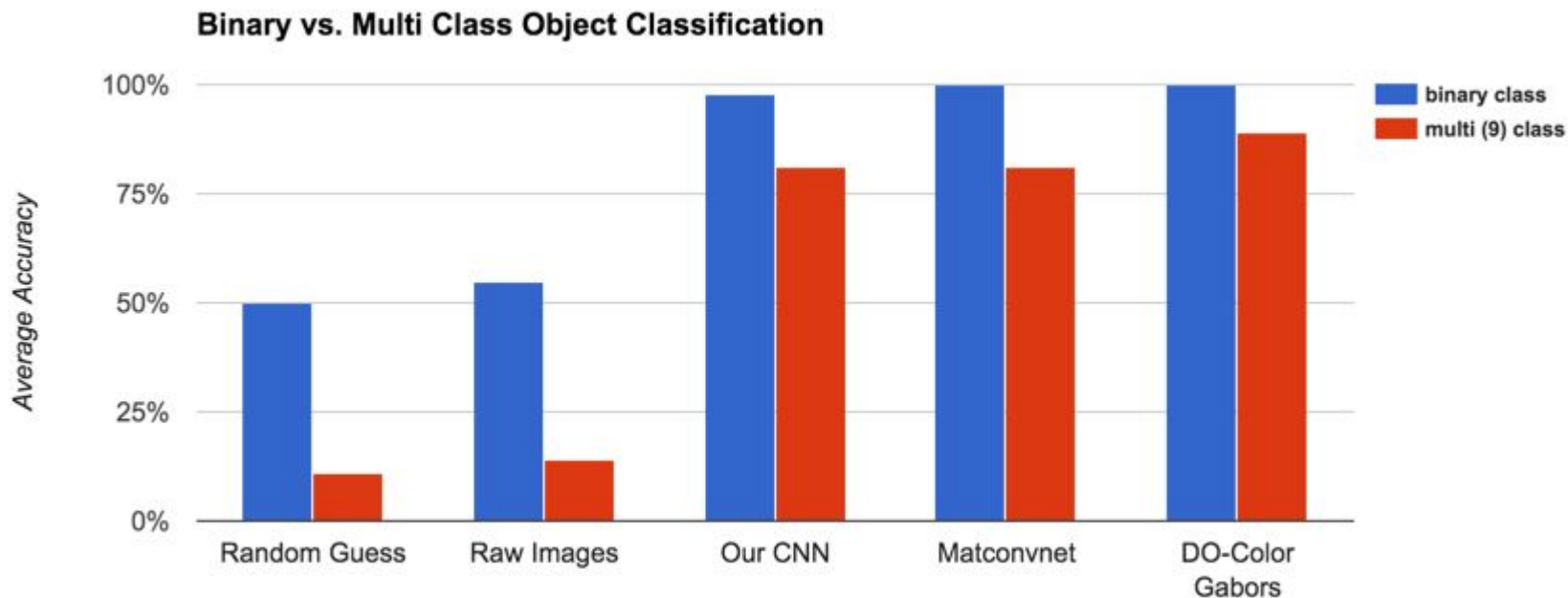
- All 9 classes - 76.0% accuracy (500 training steps)

Best convolution kernel was 3x3 pixels with 2x2 maxpooling

Far more training examples needed for our CNN



Overall Results



DeepDreams

Original



Pig Infused



Conclusion

What Worked

Well annotated dataset

Adapted homeworks worked great

Trained our own CNN

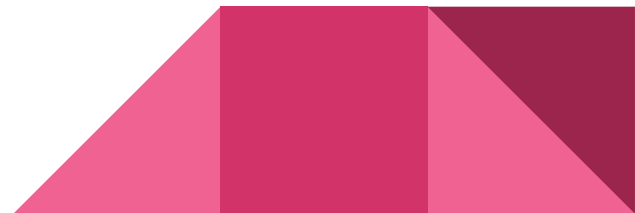
Awesome Deep Dreams pictures

What Didn't Work

Installation / Configuration were major pain points

Scene Recognition with R-CNN

Tensor Flow image library released *yesterday*

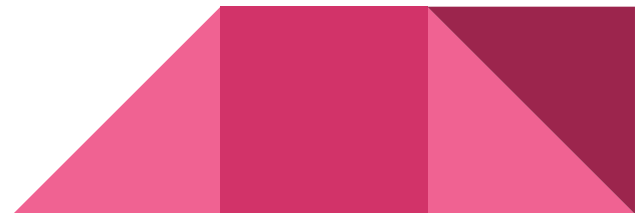


Future Work

R-CNNs for scene recognition

Deep Dreams texture pack

Apply our CNN to Deep Dreams



Questions?



Contributions - Justin

Week 1

- Brainstorming and outlining the project (3 hours)
- Minecraft installation and recording setup (2-3 hours)
- Recording data (2 hours)
- Document review (2 hours)

Week 2

- Compiling data (2-3 hours)
- Wrote and tested the video converter and data processing scripts (4 hours)
- Installed TensorFlow and went through some examples of using it (6-8 hours)
- Working with others to determine the best classification methods (3 hours)

Week 3

- Wrote data import and processing script for Python (4 hours)
- Recording more data (2 hours)
- Refining the video conversion and data processing steps (3-4 hours)
- Training custom CNN with TensorFlow using various image sizes, weights, kernel sizes and image distortions (10 hours setup, running took a very long time without GPU support, 30+ hours runtime)
- Wrote and formatted the overall presentation and the data and CNN sections (5 hours)

Contributions - Miguel

Week 1

- Brainstorming (2-3 hours)
- Minecraft Installation / Recording Setup (2 hours)
- Recording / Refining Recorded Data (4 hours)
- Biome classifier, rgb + gray (MATLAB) (2 hours)

Week 2

- More minecraft recordings / data manipulation (3 hours)
- Installed caffe + CUDA successfully in Linux. (This, I believe, is a feat on its own) (15-20 hours)
- Attempt to install RCNN + opencv, got close, but no dice (6 hours)
- Image resize script on the ~2000 images (2 hours)

Week 3

- Multiclass object classifier, rgb + gray, animals + with anti-objects (MATLAB) (4 hours)
- Deepdreams/Inceptionism. (8 hours)
 - Starting from real images, minecraft, and random noises. (python)
 - With and without tweaking its optimization objective

Contributions - Pato

Week 1

- Reading literature for other project ideas (DeepDreams + Video Magnification): 5 hours
- Brainstorming: 2-3 hours
- Recording Setup / Modifying Minecraft for appropriate filming conditions: 3 hours
- Recording : 1 hour
- Writing planned methodology for the project: 3 hours
- Reading documentation on Caffe's DNNs: 3 hours

Week 2

- Updating our methodology write-up: 2 hours
- Reading literature on semantic segmentation: 4 hours
- Compiling findings / deciding on an approach: 2 hours
- Attempt on installing Caffe's dependencies on Windows: 18 hours
- Attempt on getting Caffe to work in the CIT: 4 hours

Week 3

- Attempt on getting rcnn to work in Caffe at the CIT: 4 hours
- DeepDreams with Caffe at the CIT: 1 hour
- More data collection: 1 hour
- Second attempt to get Caffe to work on Windows (highly not recommended)
- Presentation: 3 hours

Contributions - Mason

Week 1

- Brainstorming / outline : 2-3 hours
- Minecraft Installation / Recording Setup / Modifying Minecraft for appropriate filming conditions: 3 hours
- Recording / Refining Recorded Data: 4 hours
- Document / Write up: 3 hours

Week 2

- Document / Writeup refined from week 1, and added week 2 accomplishments and contributions: 4 hours
- Caffe & sk-image/scipy/numpy tutorials: 6-7 hours
- Caffe attempt on CS machines (got denied permissions on department machines for pre-installed version, tried to install on department machines from scratch because of it, couldn't get matlab or python build to run): 4 hours

Week 3

- Caffe attempt on personal computer. Got every dependency to go through and load except for hdf5, which prohibited progress entirely. (after downloading/configuring cuda, searching through documentation on various errors: 20 hours)
- Caffe attempt 2 on cs department machines, got caffe up and running but could not get r-cnn to work on the department machines (permissions denied for added libraries on top of caffe necessary to run r-cnn): 4 hours
- Unfiltered, Sobel Filters & Gabor Filters code in Matlab & Python: 10-12 hours
- Results aggregation & Presentation: 5 hours

Extra Work

What we couldn't fit in slides

1. Sobel Filters for object recognition
2. Regular gabor filters (space invariant / frequency invariant).
3. Alternate TensorFlow CNN constructions



Object Classification: Sobel Filters

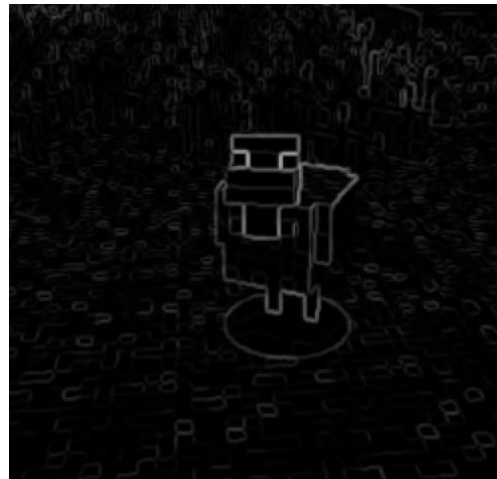
Gradient Based Filters

Only work on grayscale

Single Class Accuracy: ~55-60%

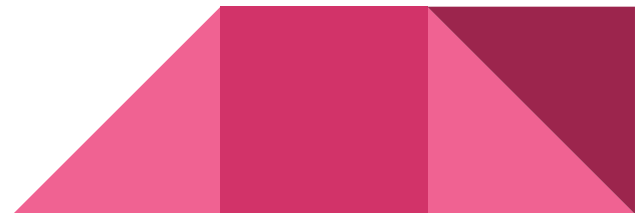
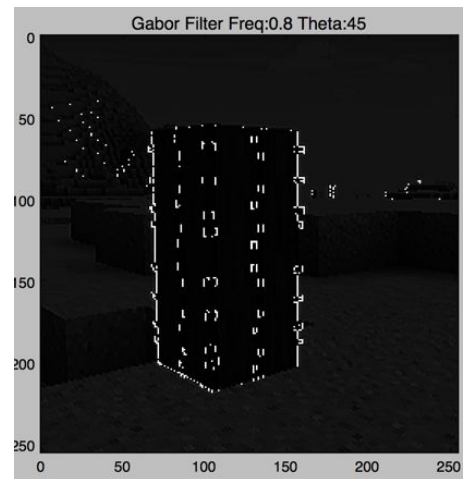
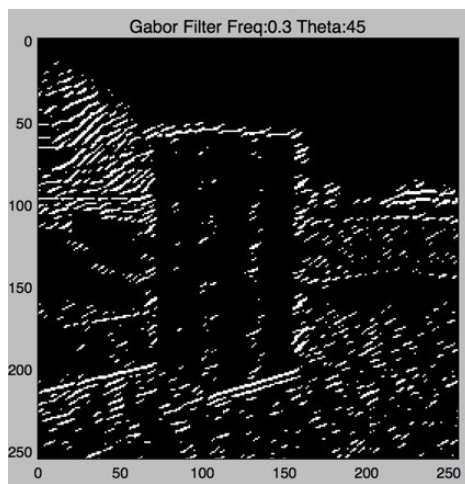
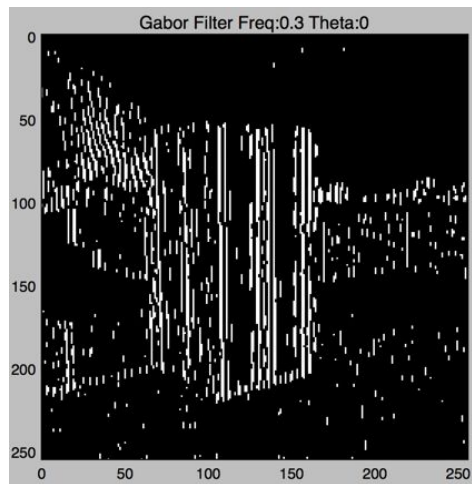
Multi (7) Class Accuracy: ~18%

More info in notes below



Gabor Filters

- Looks great
- Tuned on frequency and orientation
- No difference in accuracies (same as unfiltered)



Alternate CNN Constructions in TensorFlow

I started with a shallower CNN with only two convolutional layers and a single random dropout layer at the top before softmax. This network trained faster, but yielded worse accuracy (86% accuracy with cactus vs desert no object).

I also tried different optimizer functions, batch sizes, and keep probabilities on the dropout layers, but these would all take up too much space to describe compared to the network that performed the best.

